Zbirke podatkov 2

Komponente jezika SQL

SQL jezik - delitev

- DDL - data definition language
  - definicija in spreminjanje podatkovnih struktur (metadata): baza, tabele, pogledi, itn.
- DML - data manipulation language
  - upravljanje s podatki: dodajanje, spreminjanje, izbiranje, itn.
- DCL - data control language
  - kontrola dostopa do posameznih objektov: dovoljenja, itn.

SQL jezik - DDL

- DDL ukazi so:
  - CREATE TABLE – kreiranje nove tabele
  - ALTER TABLE – spreminjanje obstoječe tabele
  - DROP TABLE – brisanje tabele

Kreiranje tabele - CREATE TABLE

Tabela se v bazi kreira takoj po uspešno izvedenem CREATE ukazu.

Splošna sintaksa za kreiranje tabele je:

```
CREATE TABLE [shema.] NazivTabele

      (Stolpec1 TipPodatka [CONSTRAINT
naziv_omejitve] TipOmejitve,

      (Stolpec2 TipPodatka [CONSTRAINT
naziv_omejitve] TipOmejitve,...

      [CONSTRAINT naziv_omejitve] TipOmejitve
(Stolpec, ... ), ... );
```

Definicija stolpcev/polj

- Kaj je potrebno definirati?
  - naziv stolpca (ang *column name*) – obvezno!
  - podatkovni tip (ang. *data type*) in velikost – obvezno!
  - omejitve (ang. *column constraints*) - neobvezno:
    - primarni ključ (*primary key*),
    - obvezna vrednost (*null/not null*),
    - privzeta vrednost (*default*),
    - enolična vrednost (*unique*),
    - vrednost iz zaloge vrednosti/kontrola (*check*)

Podatkovni tipi

TIP	OPIS
CHAR (<velikost>)	za shranjevanje alfanumeričnih znakov. Minimum je 1. max je 2000.
VARCHAR(<velikost>)	za shranjevanje variabilnega števila znakov – minimum je 1 in maksimum je 4000.
INTEGER(p) **FLOAT(p, s)** **DOUBLE(p, s)**	za shranjevanje števil (fiksna in plavajoča vejica). Max. natančnost (p) in/ali decimalni del (s) je 38.
DATE	za shranjevanje datuma in časa. Datum je fiksne velikosti (7-bytov). Možnost različnih formatov (privzeto DD-MON-YY). Možnost velikega števila različnih formatov.
LONG	za shranjevanje do 2GB znakov. Samo en stolpec v tabeli je lahko tega tipa. Ne more biti indeksiran.
RAW (<velikost>)	uporablja se za shranjevanje binarnih podatkov, kot so grafika, zvok in podobno. Max. velikost je 2000 bytov. Lahko se indeksira.
LONGRAW	podobna LONG tipu. Za shranjevanje binarnih podatkov.
CLOB	znakovni veliki objekti. Max. velikost je 4 GB.
BLOB	binarni veliki objekti. Max. velikost je 4 GB.
BFILE	binarne datoteke. Vsebuje lokator do velike binarne datoteke shranjene zunaj baze. Omogoča I/O tok za dostop do zunanje LOB, ki se nahaja na baznem strežniku. Max. velikost je 4GB.

Tipi podatkov: VARCHAR

- VARCHAR je znakovni tip, ki omogoča shranjevanje nizov znakov (alfanumerični) ***spremenljive dolžine*** (do določenega maksimuma, ki je določen z velikostjo polja).
- Velikost je specificirana znotraj oklepajev, npr. VARCHAR(20).
- Če je podatek manjši od specificirane velikosti, se v stolpcu shranjuje samo do te velikosti – preostali prazni znaki se ne dodajajo originalnem podatku.
- VARCHAR je najbolj primeren tip za vrednosti, ki nimajo fiksne velikosti.

Tipi podatkov: CHAR

- CHAR je znakovni tip, ki omogoča shranjevanje nizov znakov (alfanumerični) *nespremenljive dolžine*.

- CHAR tip bolj učinkovito izkorišča prostor v RDBMS in obdeluje podatke hitreje, kot VARCHAR tip.

Tipi podatkov: NUMERIČNI

- INTEGER je celo število brez decimalnega dela.

- NUMBER (FLOAT, DOUBLE) se uporablja za shranjevanje negativnih, pozitivnih, celih števil ter decimalnih števil z fiksno in plavajočo vejico.

- Ko se uporablja tip NUMBER, je potrebno navesti **velikost** (*ang. precision*) in **število decimalnih mest** (*ang. scale*):
  - velikost je skupno število števk – skupaj na levi in desni strani decimalne vejice.
  - število decimalnih mest je skupno število števk na desni strani decimalne vejice.

Primer kreiranja tabele

- Primer kreiranja tabele STUDENTI:

```
CREATE TABLE STUDENTI
  (STUDENT_ID    CHAR (5),
   PRIIMEK       VARCHAR (15) NOT NULL,
   IME           VARCHAR (15) NOT NULL,
   ULICA         VARCHAR (25),
   MESTO         VARCHAR (15),
   POSTA         CHAR (4) DEFAULT '1000',
   ZACETEK       CHAR (4),
   DAT_ROJ       DATE,
   PROGRAM_ID    INTEGER (3),
   SMER_ID       INTEGER (3),
   TELEFON       CHAR (10),
   CONSTRAINT STUDENT_STUDENT_ID_PK PRIMARY KEY
  (STUDENT_ID));
```

Standardni podatkovni tipi

- Standard: ISO/IEC 9075
- Podatkovni tipi se v določeni meri razlikujejo med posameznimi SUPB (RDBMS)
- Primera (reference za podatkovne tipe):
  - Access 2007:

    http://msdn.microsoft.com/en-us/library/bb208866.aspx
  - SQL Server 2005:

    http://msdn.microsoft.com/en-us/library/ms187752(SQL.90).aspx

Tipi omejitev

Obstajata dve vrsti omejitev (*ang. constraints*):

1. Integritetne omejitve *(ang. integrity constraints)*: obsegajo primarni ključ in tuje ključe na tabeli ter primarni ključ na tabeli na katero se tuji ključ referencira.

2. Vrednostne omejitve *(ang. value constraints)*: definirajo ali so dovoljene prazne vrednosti v stolpcih tabel, ali so potrebne edinstvene vrednosti in ali je neka množica vrednosti v stolpcu dovoljena.

Imenovanje omejitev

- Splošno pravilo za imenovanje omejitev je:

```
<naziv tabele>_<stolpec>_<tip omejitve>
```

- *table name* je naziv tabele na kateri se definira omejitev,
- *column name* je naziv stolpca na katerega se nanaša omejitev,
- in *constraint type* je okrajšava, ki se uporablja za identificiranje tipa omejitve.

Imenovanje omejitev

- Na primer, omejitev `ZAPOSLENI_ODDELEK_ID_FK` pomeni:
  - omejitev v tabeli ZAPOSLENI na stolpcu ODDELEK_ID.
  - omejitev je tipa tuji ključ (FK).
  - tuji ključ se povezuje na tabelo z oddelki preko polja ODDELEK_ID.
- Na primer, omejitev `ODDELKI_ODDELEK_ID_PK` pomeni:
  - omejitev na tabeli ODDELKI na stolpcu ODDELEK_ID.
  - omejitev je tipa primarni ključ (PK).

Popularne okrajšave za omejitve

Tip	Okrajšava	
primarni ključ	PK	*ang. primary key*
tuji ključ	FK	*ang. foreign key*
unikatni ključ	UK	*ang. unique key*
vrednost iz zaloge vrednosti - kontrola	CK	*ang. check*
obvezna vrednost	NN	*ang. not null*

Definiranje omejitev (constraints)

- Omejitev (*ang. constraint*) se lahko kreira v času:
  - kreiranja tabele (ukaz `CREATE TABLE` ...)
  - lahko se doda naknadno (ukaz `ALTER TABLE` ...)

- Lahko se definira na dveh nivojih:
  - nivo stolpca/polja (*ang. column level*)
  - nivo tabele (*ang. table level*)

Omejitve - nivo stolpca

- Omejitev na nivoju stolpca se nanaša samo na ustrezni stolpec in se definira skupaj s samim stolpcem.

- Vsaka omejitev se lahko definira na nivoju stolpca razen tujega ključa in sestavljenega primarnega ključa.

Sintaksa:

Stolpec TipPodatka **[CONSTRAINT** ***naziv_omejitve*****]**
TipOmejitve

Primer:

```
STAVBA VARCHAR(7) CONSTRAINT lokacija_stavbe_NN NOT NULL
```

Omejitve - nivo tabele

- Omejitve na nivoju tabele se nanašajo na enega ali več stolpcev in se definirajo ločeno od definicije stolpcev.
- Ponavadi se navedejo *po* definiciji stolpcev.
- Vse omejitve se lahko definirajo na nivoju tabele razen omejitve za obvezno polje (NOT NULL constraint).

Sintaksa:

```
[CONSTRAINT naziv_omejitve] Tip_omejitve (Stolpec,...),
```

Primer:

```
CONSTRAINT LOKACIJA_ID_PK PRIMARY KEY(LOKACIJA_ID);
```

Primarni ključ (primary key constraint)

- Primarni ključ je znan kot *entitetna integritetna omejitev* (*ang. entity integrity constraint*)

- Kreira primarni ključ za tabelo. Tabela lahko ima *samo en* primarni ključ.

- Če tabela uporablja **sestavljeni ključ** (več kot en stolpec), potem se takšen ključ lahko definira samo na nivoju tabele.

Primarni ključ – primera

- Na nivoju stolpca je definicija naslednja:

```
STM_ID NUMBER (2) CONSTRAINT STM_STM_ID_PK PRIMARY KEY
```

- Na nivoju tabele je definicija naslednja:

```
CONSTRAINT STM_STM_ID_PK PRIMARY KEY(STM_ID),
```

Tuji ključ (foreign key constraint)

- **Tuji ključ** je znan tudi kot omejitev referenčne integritete (*ang. referential integrity constraint*).

- Uporablja stolpec ali stolpce, kot tuje ključe ter določa povezavo med primarnim ključem *iste* ali *druge* tabele.

Tuji ključ

- Za vzpostavitev tujega ključa na tabeli mora obstajati *druga* (referencirana) tabela in *njen primarni ključ*!

- Stolpca za tuji ključ in referencirani primarni ključ v drugi tabeli ni treba da imata enaki imeni vendar vrednost tujega ključa mora ustrezati vrednosti v nadrejeni (referencirani tabeli) ali biti NULL!

Tuji ključ – primer

- Tuji ključ se kreira samo na nivoju tabele!

- Primer: povezujemo tabelo STUDENT in tabelo PROGRAMI preko polja PROGRAM_ID (student je vpisan na program PROGRAM_ID):

```
CONSTRAINT STUDENT_PROGRAM_ID_FK FOREIGN KEY(PROGRAM_ID)
REFERENCES PROGRAMI (PROGRAM_ID),
```

Obvezno polje (NOT NULL constraint)

- Omejitev za obvezno polje zagotavlja, da bo ustrezni stolpec v tabeli vedno imel vrednost.
- Presledek ali 0 (kot numerična vrednost) nista null-vrednosti!
- Omejitev se definira samo na nivoju stolpca:

```
Naziv VARCHAR(15) CONSTRAINT sola_naziv_NN NOT NULL,
```

Unikatni ključ (unique key constraint)

- Unikatni ključ zahteva, da so vrednosti v ustreznem stolpcu ali skupini stolpcev, edinstvene (*ista vrednost se lahko pojavi samo enkrat*).

Na nivoju stolpca je omejitev definirana kot:

```
DeptName VARCHAR(12) CONSTRAINT
dept_deptname_uk UNIQUE,
```

Na nivoju tabele se omejitev definira kot:

```
CONSTRAINT dept_deptname_uk UNIQUE(DeptName),
```

Kontrole (check constraint)

- Kontrola (*CHECK constraint*) definira pogoj, ki mora biti izpolnjen na vsakem zapisu v tabeli.

Na nivoju stolpca se omejitev definira kot:

```
STM_ID NUMBER(2) CONSTRAINT OE_STM_ID_CC
    CHECK((STM_ID >= 10) AND (STM_ID <= 99)),
```

Na nivoju tabele se omejitev definira kot:

```
CONSTRAINT OE_STM_ID_CC
    CHECK((STM_ID >= 10) AND (STM_ID <= 99)),
```

Spreminjanje tabele - ALTER TABLE

- Z ukazom ALTER TABLE lahko spreminjamo podatkovne tabele
- Spremembe vključujejo:
  - stolpci/polja (*columns*):
    - dodajanje (*add*),
    - spreminjanje (*modify*),
    - brisanje (*drop*)
  - omejitve (*constraints*):
    - dodajanje (*add*),
    - spreminjanje (*modify*),
    - brisanje (*drop*)

```
ALTER TABLE NazivTabele [SPREMEMBE]
```

Dodajanje novega stolpca v tabelo

- Novi stolpec se v obstoječo tabelo doda z ukazom:

```
ALTER TABLE NazivTabele ADD NazivStolpca TipPodatka;
```

Primer:

```
ALTER TABLE student ADD DAVCNA_STEVILKA VARCHAR(10);
```

Sprememba obstoječega stolpca

- Obstoječi stolpec se v obstoječo tabelo doda z ukazom:

```
ALTER TABLE NazivTabele MODIFY
NazivStolpca NoviTipPodatka;
```

Primer:

```
ALTER TABLE student MODIFY DAVCNA_STEVILKA VARCHAR(8);
```

Brisanje obstoječega stolpca iz tabele

- Obstoječi stolpec se lahko odstrani iz tabele z ukazom:

```
ALTER TABLE NazivTabele DROP COLUMN
NazivStolpca NoviTipPodatka;
```

Primer:

```
ALTER TABLE student DROP COLUMN DAVCNA_STEVILKA;
```

Dodajanje omejitve

- Obstoječi stolpec se lahko odstrani iz tabele z ukazom:

```
ALTER TABLE NazivTabele ADD [CONSTRAINT
naziv_omejitve Tip_omejitve (stolpec, ...)
```

ALTER TABLE - primeri

- Primer:

dodajanje tujega ključa:

```
ALTER TABLE STUDENT
   ADD CONSTRAINT student_servis_ID_FK
   FOREIGN KEY(SERVIS_ID)
   REFERENCES SSERVISI(SERVIS_ID) ON UPDATE
  CASCADE;
```

Brisanje tabele - DROP TABLE

```
DROP TABLE NazivTabele [RESTRICT|CASCADE];
```

POZOR: DROP ukaz za vedno odstrani tabelo – strukturo in podatke!

Ima dve opciji:

- **CASCADE**: določa, da se pri vseh povezav preko tujih ključev, ki so povzročene z brisanjem tabele, bodo izbrisali ustrezni zapisi v povezani tabeli.
- **RESTRICT**: blokira brisanje tabele, če obstajajo zapisi v tabeli povezani preko tujih ključev.

Indeksi

- Bazni objekti namenjeni hitrem iskanju podatkov
- Za primarni ključ se naredi avtomatično
- Ostale naredimo po potrebi:

```
CREATE [UNIQUE] [ASC] | DESC] INDEX ime_indeksa ON
ime_tabele (atribut1, atribut2, ..);
```

- Brisanje sekundarnih indeksov:

```
DROP INDEX ime_indeksa;
```